SVRE: NEW METHOD FOR TRAINING GANs

Gauthier Gidel
Mila, Université de Montréal
Research intern at Element AI

Generative Modeling and Model-Based Reasoning for Robotics and AI Workshop
June 14, 2019
Reducing Noise in GAN Training with Variance Reduced ExtraGradient

Tatjana Chavdarova *

Gauthier Gidel *

François Fleuret

Simon Lacoste-Julien

* Equal contribution
GENERATIVE ADVERSARIAL NETWORKS
[Goodfellow et al., 2014]
CHALLENGES

- Standard supervised learning:

\[
\min_{\theta} \mathcal{L}(\theta)
\]

- GANs: Hard (different) optimization problem: minimax.
CHALLENGES

- Standard supervised learning:

$$\min_{\theta} \mathcal{L}(\theta)$$

- GANs: Hard (different) optimization problem: minimax.

$$\min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D)$$

Image source: Vaishnavh Nagarajan
“Noise”: Noisy Gradient Estimates Due to Stochasticity

- Using sub-samples (mini-batches) of the full dataset to update the parameters
- Variance Reduced (VR) Gradient: optimization methods that reduce such noise

Minimization: Single-objective

- Batch method direction
- Stochastic method direction: noisy
VARIANCE REDUCTION—MOTIVATION FOR GAMES

- **Intuitively:** **Minimization** vs. **Game** (Noise from Stochastic gradient)

- **Empirically:** **BigGAN**—“Increased batch size significantly improves performances”

- **To sum up, two issues:**

  Minimization
  “approximately” the right direction

  Game
  Direction with noise can be “bad”
Variance Reduction—Motivation for Games

- Intuitively: Minimization Vs. Game (Noise from Stochastic Gradient)
- Empirically: BigGAN—“Increased batch size significantly improves performances”
- To sum up, two issues:

Brock et al. [2018] report a relative improvement of 46% of the Inception Score metric [Salimans et al., 2016] on ImageNet if the mini-batch size is increased 8-fold.
Variance Reduction—Motivation for Games

- Intuitively: Minimization Vs. Game (Noise from Stochastic Gradient)
- Empirically: BigGAN—“Increased batch size significantly improves performances”

- To sum up, two issues:

  - Adversarial aspect from min-max → Extragradient.
  - Noise from stochastic gradient → Variance Reduction.
EXTRAGRADIENT
Two players $\theta$, $\varphi$. Idea: perform a “Lookahead step”

Extrapolation:
\[
\begin{align*}
\theta_{t+1/2} &= \theta_t - \eta \nabla_\theta \mathcal{L}_G(\theta_t, \varphi_t) \\
\varphi_{t+1/2} &= \varphi_t - \eta \nabla_\varphi \mathcal{L}_D(\theta_t, \varphi_t)
\end{align*}
\]

Update:
\[
\begin{align*}
\theta_{t+1} &= \theta_t - \eta \nabla_\theta \mathcal{L}_G(\theta_{t+1/2}, \varphi_{t+1/2}) \\
\varphi_{t+1} &= \varphi_t - \eta \nabla_\varphi \mathcal{L}_D(\theta_{t+1/2}, \varphi_{t+1/2})
\end{align*}
\]
VARIANCE REDUCED GRADIENT METHODS
VARiance REDuced ESTimate OF THE GRADient

Based on Finite sum assumption:

\[ \frac{1}{n} \sum_{i=1}^{n} L(x_i, \omega), \]

Epoch based algorithm:
- Save the full gradient \( \frac{1}{n} \sum_i \nabla L(x_i, \omega^S) \) and the snapshot \( \omega^S \).
- For one epoch use the update rule:

\[
\omega \leftarrow \omega - \eta \left[ \nabla L(x_i, \omega) + \frac{1}{n} \sum_i \nabla L(x_i, \omega^S) - \nabla L(x_i, \omega^S) \right]
\]

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If \( \omega^S \) is close to \( \omega \) \( \rightarrow \) close to full batch gradient \( \rightarrow \) small variance.
- Full batch gradient expensive but tractable, e.g., compute it once per pass.

Gauthier Gidel  Generative Modeling and Model-Based Reasoning for Robotics and AI Workshop
**Variance Reduced Estimate of the Gradient**

Based on Finite sum assumption:

$$\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(x_i, \omega),$$

Epoch based algorithm:
- Save the full gradient $\frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S)$ and the snapshot $\omega^S$.
- For one epoch use the update rule:

$$\omega \leftarrow \omega - \eta \left[ \nabla \mathcal{L}(x_i, \omega) + \frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S) - \nabla \mathcal{L}(x_i, \omega^S) \right]$$

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If $\omega^S$ is close to $\omega$ → close to full batch gradient → small variance.
- Full batch gradient expensive but tractable, e.g., compute it once per pass.
**Variance Reduced Estimate of the Gradient**

Based on Finite sum assumption:

\[
\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(x_i, \omega),
\]

Epoch based algorithm:
- Save the full gradient \( \frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S) \) and the snapshot \( \omega^S \).
- For one epoch use the update rule:

\[
\omega \leftarrow \omega - \eta \left[ \nabla \mathcal{L}(x_i, \omega) + \frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S) - \nabla \mathcal{L}(x_i, \omega^S) \right]
\]

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If \( \omega^S \) is close to \( \omega \) \( \rightarrow \) close to full batch gradient \( \rightarrow \) small variance.
- Full batch gradient expensive but tractable, e.g., compute it once per pass.
**Variance Reduced Estimate of the Gradient**

Based on Finite sum assumption:

\[
\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(x_i, \omega),
\]

Epoch based algorithm:

- Save the full gradient \( \frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S) \) and the snapshot \( \omega^S \).
- For one epoch use the update rule:

\[
\omega \leftarrow \omega - \eta \left[ \nabla \mathcal{L}(x_i, \omega) + \frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S) - \nabla \mathcal{L}(x_i, \omega^S) \right]
\]

- Requires 2 stochastic gradients (at the current point and at the snapshot).
- If \( \omega^S \) is close to \( \omega \) → close to full batch gradient → small variance.
- Full batch gradient expensive but tractable, e.g., compute it once per pass.
SVRE: Variance reduction + ExtrAGRAdient

Pseudo-algorithm

1. Save snapshot $\omega^S \leftarrow \omega_t$ and compute $\frac{1}{n} \sum_i \nabla L(x_i, \omega^S)$.
2. For $i$ in $1, \ldots, \text{epoch\_length}$:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at $\omega_t$.
   - Compute $\omega_{t+1}$ with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$
3. Repeat until convergence.
SVRE: Variance Reduction + Extragradient
Pseudo-algorithm

1. Save snapshot $\omega^S \leftarrow \omega_t$ and compute $\frac{1}{n} \sum_i \nabla L(x_i, \omega^S)$.

2. For $i$ in $1, \ldots, \text{epoch\_length}$:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at $\omega_t$.
   - Compute $\omega_{t+1}$ with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$

3. Repeat until convergence.
SVRE: VARIANCE REDUCTION + EXTRAGRADIENT
PSEUDO-ALGORITHM

1. Save snapshot $\omega^S \leftarrow \omega_t$ and compute $\frac{1}{n} \sum_i \nabla \mathcal{L}(x_i, \omega^S)$.
2. For $i$ in $1, \ldots, \text{epoch\_length}$:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at $\omega_t$.
   - Compute $\omega_{t+1}$ with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$
3. Repeat until convergence.
SVRE: VARIANCE REDUCTION + EXTRAGRADIENT
PSEUDO-ALGORITHM

1. Save snapshot $\omega^S \leftarrow \omega_t$ and compute $\frac{1}{n} \sum_i \nabla L(x_i, \omega^S)$.
2. For $i$ in 1, \ldots, \text{epoch\_length}:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at $\omega_t$.
   - Compute $\omega_{t+1}$ with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$
3. Repeat until convergence.
SVRE: VARIANCE REDUCTION + EXTRAGRADIENT

PSEUDO-ALGORITHM

1. Save snapshot $\omega^S \leftarrow \omega_t$ and compute $\frac{1}{n} \sum_i \nabla L(x_i, \omega^S)$.

2. For $i$ in 1, …, epoch_length:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at $\omega_t$.
   - Compute $\omega_{t+1}$ with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$

3. Repeat until convergence.
SVRE: Variance Reduction + Extragradient
Pseudo-algorithm

1. Save snapshot $\omega^S \leftarrow \omega_t$ and compute $\frac{1}{n} \sum_i \nabla L(x_i, \omega^S)$.
2. For $i$ in $1, \ldots, \text{epoch\_length}$:
   - Compute $\omega_{t+\frac{1}{2}}$ with variance reduced gradients at $\omega_t$.
   - Compute $\omega_{t+1}$ with variance reduced gradients at $\omega_{t+\frac{1}{2}}$.
   - $t \leftarrow t + 1$
3. Repeat until convergence.

SVRE yields the fastest convergence rate for strongly convex stochastic game optimization in the literature.
SVRE: EXPERIMENTS
Stochastic baseline

- **Always** diverges.
- **Many hyperparameters** \((\eta_G, \eta_D, \beta_1, \gamma, r)\).
- **If convergence** → fast
**EXPERIMENTS**

SVRE yields stable GAN optimization

---

**Stochastic baseline**

- Always diverges.
- Many hyperparameters \((\eta_G, \eta_D, \beta_1, \gamma, r)\).
- If convergence \(\rightarrow\) fast

**SVRE**

- Does not diverge.
- Fewer hyperparameters (omits \(\beta_1, \gamma, r\)).
- Slower for very deep nets.
SVRE: Takeaways
SVRE: Takeaways

- Controlling variance is more critical for games (could be reason behind success of Adam on GANs)
- **SVRE**: combines Extragradient and variance reduction.
- Best convergence rate (under some assumptions) for games.
- Good stability properties.
THANKS.

Questions?


THE GAN FRAMEWORK

EQUILIBRIUM AT $p_g = p_d$

The discriminator maximizes:

$$V(G, D) = \int_x p_d(x) \log(D(x)) \, dx + \int_z p_z(z) \log(1 - D(G(z))) \, dz$$

$$= \int_x p_d(x) \log(D(x)) + p_g(x) \log(1 - D(x)) \, dx$$

Where we used $x = G(z)$, and $p_g$ is the distribution of $x$. Hence, the optimal discriminator $D^*$ is:

$$D^*(x) = \frac{p_d(x)}{p_d(x) + p_g(x)}$$
The GAN framework

**Equilibrium at** $p_g = p_d$

The generator minimizes:

$$V(G, D^*) = \mathbb{E}_{x \sim p_d} \left[ \log D^*(x) \right] + \mathbb{E}_{x \sim p_g} \left[ \log(1 - D^*(x)) \right]$$

$$= \mathbb{E}_{x \sim p_d} \left[ \log \frac{p_d(x)}{p_d(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[ \log \frac{p_g(x)}{p_d(x) + p_g(x)} \right]$$

$$= - \log 4 + \mathbb{D}_{KL}(p_d \parallel \frac{p_d + p_g}{2}) + \mathbb{D}_{KL}(p_g \parallel \frac{p_d + p_g}{2})$$

$$= - \log 4 + 2 \cdot \mathbb{D}_{JS}(p_d \parallel p_g)$$

where we used: $\mathbb{D}_{JS}(p \parallel q) = \frac{1}{2} \mathbb{D}_{KL}(p \parallel \frac{p+q}{2}) + \frac{1}{2} \mathbb{D}_{KL}(q \parallel \frac{p+q}{2})$.

The optimum is reached when $p_g = p_d$, and the optimal value is $- \log 4$. 

Gauthier Gidel

Generative Modeling and Model-Based Reasoning for Robotics and AI Workshop

19 / 17